Project 1: Swallow Stars (action game)

Event-Driven Programming with Text Graphics

Michał Małafiejski
KAMS WETI PG (micmalaf@pg.edu.pl)

Basics of Computer Programming 2025

Epic Intro

Swallow Stars opens high above the cliffs. A restless swallow leaves her nest before dusk to gather starlit fireflies that sustain her tribe. Crosswinds, hunter shadows, and fading light turn the flight into a race.
The elders warn: once hunters lock on, they rarely relent. Bounces off the cliff buy seconds, but the sky fills fast. Quick reflexes, steady flight, and smart speed control keep her alive—and her tribe fed.

Mission: Collect the target number of stars before the timer hits zero.

Why?

- strengthen C / C++ fundamentals using procedural programming
- master text-mode graphics and real-time input handling
- practise event-driven loops with multiple moving actors
- implement configuration-driven gameplay tuned from files

How? (limitations)

- Use any ANSI/ISO compliant C/C++ compiler (Linux, macOS, Windows)
- Pick one text graphics library: `*curses.h` (Unix-like) or `conio*.h` (Windows)
- Follow procedural style with your own data structures (`struct`) and functions; avoid OOP constructs and STL containers, both of which which are forbidden
- Obey the 1024-byte character limit per function (`main()` included), excluding whitespace and comments

How? (gameplay core)

- Split the screen: sky with moving actors above, status area below
- Stars (*) spawn at top, fall at varied speeds, vanish on landing
- Swallow is always in motion; controls: `a/w/s/d`
- Change speed with `o/p`; clamp 1–5
- Swallow life force decreases when hitting by hunter type (size/color); at 0: *game over*
- Colors encode state; e.g., hunter by type, swallow by life force
- Config file: map, quotas, timers, hunters, seeds

How? (hunters)

- Hunters enter from screen borders, lock heading toward the swallow, and keep that vector (direction) with a constant speed
- Each hunter carries a countdown label of remaining bounces; reaching zero despawns it
- Upon hitting a border, each hunter either reflects perfectly or disappears
- Hunter "sprites" must span a few cells, at least these shapes 1x2, 2x1, 1x3, 3x1, or 2x2 cells, and more will be appreciated; manage clean reflections for every shape

What?

- Obligatory scope (max. 6 pts)
  1. Status area: player, level, time left, stars, life force (1 pt)
  2. Star spawning, motion, and capture detection paired with swallow movement (1 pt)
  3. Hunter spawning logic, trajectory locking, collision handling, and bounce countdowns (1 pt)
  4. Configurable text file controlling star quota, timers, spawn rates, hunter templates, damage rules, and speed bounds (1 pt)
  6. Structured code with dedicated modules for actors, physics, and I/O (2 pts)
- Optional challenges (max. 11 pts)
  8. During a level, hunter counts and initial bounce counters escalate over time (2 pts)
  9. Hunter logic: if the projected path would miss the swallow, hunter pauses briefly, then dashes to intercept (1 pt)
  10. Friendly albatross taxi that can carry the swallow to a safe zone on request (1 pt)
  11. Ranking stored in files: compute score from time used, stars collected, life remaining, and difficulty; display top N (1 pt)

What? (cont)

Scored only any *prefix* from above task list numbers, i.e. a failure to complete task number i results in not scoring subsequent tasks $i + 1, i + 2, \ldots$.

- Advanced feats (choose up to 16 pts total)
  - Boss eagle patrolling diagonals with non-constant speed, predicting the swallow's future location; never stops (2 pts)
  - Replay recording for post-flight analysis (2 pts)
  - ASCII animation: swallow flaps wings in-flight via frame alternation; stars blink or shift color as lifetime fades (2 pts)
  - Wind gust system altering velocities unpredictably yet smoothly (3 pts)
  - Nightfall fog gradually shrinking the playable airspace (3 pts)
  - Fully configurable levels in files (min. five): each level controls map size, star quota, time limit, colors, speed bounds, spawn rates, hunter types/counts, initial bounces, damage rules, scoring weights, and escalation; selectable and loaded at runtime (3 pts)

Details

- Maintain real-time loops: hunters, stars, and swallow update every tick respecting timer granularity
- Handle collisions using cell occupancy maps; ensure multi-cell hunters process every covered tile
- Persist configuration and standings using text files accessed through `FILE*` routines only
- Keep graphics legible: ASCII birds and stars must be recognizable; focus on fluid control feel
- The status line should display player identity, mission goals, and live telemetry in every frame

Details (cont)

- Each level is defined in the config: star quota, time limit, hunter templates (shape, speed, initial bounces), swallow speed bounds, and scoring weights
- Ensure constant swallow motion; when no key is pressed keep drifting using last heading and speed
- Gracefully degrade when timers expire: show cinematic over screens and write final score to ranking

Resources

- Event-driven programming notes (timers, ncurses)
  - `https://www.cs.hunter.cuny.edu/~sweiss/course_materials/unix_lecture_notes/chapter_06.pdf`
- Sample ncurses framework
  - `http://home.animima.org/gut/pp1/bird_flight.c`
  - compile: `gcc -o bird_flight bird_flight.c -lncurses`